IN THE SPECIFICATION:

At page 1 lines 10-14, through page 2 line 1, change the paragraph to read as follows:

CROSS-REFERENCES TO RELATED APPLICATIONS

This disclosure is related to the following co-pending applications, entitled:

a1

MESSAGE CONTROL SYSTEM FOR MANAGING MESSAGE RESPONSE IN A KERBEROS ENVIRONMENT, USSN 08/884,413 filed June 22, 1997 now [[allowed]] U.S. Patent 6,003,136; and SYNCHRONOUS MESSAGE CONTROL SYSTEM IN A KERBEROS DOMAIN, USSN 08/948,840 filed October 10, 1997 now U.S. Patent 6,055,639; EXPEDITED MESSAGE CONTROL FOR SYNCHRONOUS RESPONSE IN A KERBEROS DOMAIN, USSN 09/026,746 filed February 20, 1998 now U.S. Patent 6,178,920; ASYNCHRONOUS MESSAGE SYSTEM FOR MENU-ASSISTED RESOURCE CONTROL PROGRAM, USSN 08/884,418 filed June 22, 1997 now [[allowed]] U.S. Patent 6,009,175; each of which are incorporated herein by reference.

At page 7, line 3, change the paragraph to read as follows:

SUMMARY OF THE INVENTION:

a2

The present configuration provides a system and method involving the logon of [[none]] non-preauthenticated clients which use a Kerberos domain. A multiple number of clients and principals are connected through a network cloud to a Kerberos Server and also to a client server (ClearPath NX server). The Kerberos Server provides for the administration of Kerberos operations and also provides a Key Distribution Center. The client server utilizes a combination of elements which include a Kerberos Support Library, a General Security Service unit and Master Control Program which utilize a Menu-Assisted Resource Control Program and a Communication Management System (COMS) which cooperate to provide credentials or find credentials for each non-preauthenticated client in order that the client may logon to the system in order to utilize the Kerberos domain.

At page 7, line 19, change the paragraph to read as follows:

a3

The authentication of the client or principal who participates in a given Kerberos domain [[are]] is authenticated by an asynchronous response message after validation of the client's ability to participate is performed by the Kerberos Server. Communications between client and server are performed by passing various classes of messages using various protocols which work on an asynchronous basis.

At page 18, line 14, change the paragraph to read as follows:

a4

38. GSS-cred-handle - A credential handle. A GSS handle is of type DOUBLE and is represented by two words. The first word of the handle has three components containing important pieces of information about the handle. The second word is unused. The three key components of a handle are its type, index and qualifier. The type of a handle can have three possible values, name-handle, and credential handle or a context handle. This is stored in bits 43 through 40 of the first word. The index of a handle represents the index in the GSS internal data set. This is stored in bits 39 through 20 of the first word. The qualifier represents a random number that makes the handle unique. This random number is generated while creating the handle. This is stored in bits 19 through 0 of the first word.

At page 19, line 17, change the paragraph to read as follows:

a5

43. Telnet - A standardized protocol that logically connects a terminal or terminal emulator running in a workstation to a server. After successfully connecting to the server, [[that]] the client may directly issue commands to the server that the server interprets and responds to.

At page 24, line 32, change the paragraph to read as follows:

The directives process then operates at marker 301 to contact the Kerberos Support Library 34. Now assuming that the system is operating on a synchronous message basis, the Kerberos Support Library 34 will operate at marker 202 requesting service on behalf of the MARC program 40. During this service request all previous process within this environment will wait for the service response from the Kerberos [[Service]] Server 20. Following marker 203 back to the Kerberos Support Library 34 the response is returned. The Kerberos Support Library 34 will operate at marker 102 to send the Kerberos response to the COMS program 103, which then at marker 104, will contact the MARC program 40 which will then use the marker 105 to reconnect to the COMS program 103 with the processed message response which at marker 106 will be conveyed to the multiplexer 5x, and thence at marker 10x conveyed back to the client 10.

a6

At page 26, line 4, change the paragraph to read as follows:

a7

Now assuming that the command is a "synchronous command", then the Kerberos Support Library will operate at marker 202 requesting service on behalf of the MARC program 40. During this service request all previous process within this environment will wait for the service response from the Kerberos [[Service]] Server 20. Following marker 203 back to the Kerberos Support Library 34, the response is returned. The Kerberos Support Library 34 will operate at marker 204 to send the Kerberos response over to the COMS program 205, which then at marker 208 will convey the Kerberos message response to the multiplexer 5x, which at marker 10x will convey the synchronous message response back to the client 10.

At page 32, line 1, change the paragraph to read as follows:

a8

There are several functions that are provided by the MCP 60, but the major functional concerns are those which involve the passing of an asynchronous message in addition to handling the queue management 62 functions and management of the UDP port 15 in the principal 13. (ClearPath NX Server).

At page 32, line 7, change the paragraph to read as follows:

a9

The client 10 enters his communications via the communication connector 18 and bus 18c over into the ports 52 and then talks through the Telnet unit 48 and the station transfer group 46 designated in block 50c. These units in turn call the COMS 42 which in turn talks or communicates with MARC 40 (Menu Assisted System Resource Control program). Both COMS and MARC are Unisys A-Series computer system functions described in the attached Glossary.

At page 32, line 19, change the paragraph to read as follows:

a10

The function of the Key Distribution Center 22 (KDC) is to act as a trusted third party authority that holds the secret keys for all the principals in a realm. The KDC provides two services designated AS and TGS. "AS" denotes Authentication Service (AS): i.e., A service of KDC that verifies the authenticity of a principal. From "CyberSAFE Challenger Administrator's Guide" Version 5.2.5/April, 1995.

At page 33, line 30, change the paragraph to read as follows:

a11

Within the principal 13, Fig. 3, there is seen the port's interfaces 52 which connect to the combination unit 50c designated as Telnet/HLCN/Station Transfer. The HLCN refers to the high-level communication network. The combination module 50c then connects to the COMS 42 programs (Communications Management System) and also the MARC programs 40 (Menu Assisted Resource Control programs). Then as seen, the MARC programs connect to the Kerberos Support Library 34. Further in Fig. 3, the Master Control Program MCP 60 is seen connected to the ports 52, the combination module 50c, the COMS program 42 and the MARC program 40 and also the Kerberos Support Library 34. Each of the modules are connected to the MCP 60 (Master Control Program).

At page 43, line 5, change the paragraph to read as follows:

a12

While the earlier provided systems shown in Fig.[[5A]] 5 required the time-consuming factor of calling the Kerberos Server which was then required to access the Kerberos Support Library after which the Kerberos Support Library had to respond, it will be seen that under the presently disclosed improved system, there is no longer any need for the double jump action of accessing both the Kerberos Server and the Kerberos Support Library since under the improved system, the Kerberos Support Library can provide an immediate synchronous response without the need to access the Kerberos Server 20. This will be seen in connection with Fig. 11 which is a simplified description of Fig. 1.

At page 48, line 7, change the paragraph to read as follows:

a13

Fig. 9 (9A, 9B, 9C) indicates word formats on the ClearPath NX Server and shows the MARC message layout, the MCS message layout, and the message display format. A word (Fig. 13) consists of forty-eight data bits and three leading control bits which define the type of word. The control bits for words shown in Fig. 13 are all zero; making the control value zero. Fig. 14 is a conceptual diagram of a word array that is passed from process to process. The words as marked in Fig. 9 are represented by Fig. 13 and can be referenced as such. The passing of messages asynchronously uses a message array containing a header and message data. In this particular instance the data is being passed from the Kerberos Support Library, KSL, Fig. 3 (34) to MARC, Fig. 3 (40).

At page 51 lines 23-26, though page 52 line 1, change the paragraph to read as follows:

Independent of process P1 (Fig. [[12]] 15) on the Kerberos Server 20, process P2 (in Server 13) is the initialization process of the Kerberos Support Library 40 (Fig. [[12]] 2) resident on the ClearPath NX Server 13. Process P2 functions in a similar method to that of process P1. Process P2 executes a read R2A on the file F2. The read result R2B from file F2 returns information about any principal(s) it had been previously been made aware of, along with the "name" of the realm for which it is a member and any other configuration information. Once process P2 has completed the read operation it processes off to a dependent task P3. Process P3 initiates an update request for realm and principal information. This request is a call C3 to the Kerberos Server 20. Process P4 which, if not already initiated, is processed off. The incoming call [[C4]] C3 indicates to process P4 that an update is being requested. Process P4 executes a read R4B of file F1. The read result R4A is returned to process P4. Process P4 packages the information using a shared mutually agreed-upon protocol and initiates a return C4 back to process P3 which has been waiting. Process P3 executes a Write R3 to file F2. Upon completion of this task, process P3 notifies process P2 of the success or failure in obtaining current information from the Kerberos Server 20. Process P3 now terminates regardless of the outcome of the update. If the update process P3 returned a result indicating failure, process P2 waits a predetermined period of time. After that time period has expired the above process P3 is repeated. This continues until process P2 obtains a successful result and is made "available" to perform Kerberos-related functions. If

14

a14

the information was successfully returned, process P2 finishes its initialization by making the ClearPath NX Server 13 "available" to any principal requiring service. At this point in time both files F1 and F2 on the Kerberos Server 20 and the ClearPath NX Server 13 share in a state of synchronicity.

At page 52 lines 24-29, though page 53 lines 2-3, change the paragraph to read as follows:

Once files F1 and F2 have been synchronized (Fig. 15), any requests received by the ClearPath Server 13 can be responded to directly. In the event that a change has occurred on the Kerberos Server 20, the change is noted and file [[F1]] F2 is updated. A corresponding change is noted and file F1 is updated. A corresponding change is necessary on the ClearPath NX Server 13. To accomplish this "update" the Kerberos Server 20 initiates the process P5. Process P5 processes changes received by the Kerberos Server 20. As part of the update process, process P5 initiates call C5 to the ClearPath NX Server 13. The ClearPath NX Server 13 initiates process P6 which in turn performs a "Write" operation R6 which updates file F2. Once again at this point files F1 and F2 are again synchronized. Process P7 handles requests made via communication line 14i, (Fig. [[12]] 15). When a service is requested, the Kerberos Support Library [[40]] 34 initiates process P7. Process P7 starts a Read R7A from file F2. The Read result R7B returns the requested information. Process P7 is then able to return a message in response to the clients' request via communications line 14o. The time-saving shown here is that process P7 has a single read/process to return the result to the client. Without initial and event-driven updates, the processes P3, P5, P6 would have to be performed while the client waited. This is no longer the case, and a fast immediate response can now be effectuated to provide the appropriate response to the client.

At page 55, line 17, change the paragraph to read as follows:

a16

(11) The Kerberos Support Library 34 determines that the response is to be asynchronous (Y=yes). At this stage, the KSL 34 must obtain additional information from the immediate requester (MARC 40) about the originator. The KSL (ii, Fig. 6A) must also inform MARC the response will be returned asynchronously. Fig. 4 indicates the block "C" at position (f2) which indicates the subsequent sequences which will be described in Figs. 6A, 6B and 7A, 7B. The following steps 12 through 34 provide an initial generalized summary of these actions.

At page 55, line 24, change the paragraph to read as follows:

a17

(12) The Kerberos Support Library 34 requests (vii, Fig. 6A) the client-originator information and builds a request to be sent to the Kerberos Server 20. In addition, it builds a message (in clear text form for display) which message can be used or discarded by the originator. The message states that the response to the Kerberos command which was entered will be returned "asynchronously" as an unsolicited message.

At page 60, line 13, change the paragraph to read as follows:

a 18

This processing function then proceeds to position (xv) Fig. 6B where COMS then notes the queue event and passes the message on to MARC. Then at position (xvi), the MARC process receives the unsolicited message in the control queue Fig. 10. At position (xvi), it is seen that the MARC program receives the unsolicited message in a queue.

At page 60, line 35, change the paragraph to read as follows:

a19

The process of "receives" is an instance again of moving data from one environment to another environment using a queue. The data descriptor which points to the data in an area of memory is passed from one process to a different process. The MARC program receives the MSG (data descriptor) from COMS, and then MARC will then process this message (xxi) Fig. 6B which will ultimately be passed back to COMS for delivery shown at position (xxii). At position (xxii), it is seen that COMS receives the message to be delivered to the original requester. Here, COMS receives the message from MARC. Then through a series of procedure calls, the message is eventually delivered to the appropriate transport for delivery at the original client-requester 10.

At page 61, line 15, change the paragraph to read as follows:

a20

Thus, at position step (xviii), MARC verifies to see whether the original requester is still a valid requester, after which at position (xix), a check is made at the decision tree to determine whether the particular station is still valid. If the station is valid, Y = Yes, then at position (xx) a "valid return" signal is sent[[,]] to location step (xvi), where MARC receives the unsolicited message in a control queue, at which time on channel 65, at position step (xxi), MARC converts the encoded station information into a COMS message format. This is sent via position (xxii) whereby COMS receives the message to be delivered to the original requester at the terminal 10.

At page 62, line 31, change the paragraph to read as follows:

a21

Now MARC maps the dialogue number to the Transaction_ID and returns the dialogue number to the Kerberos Support Library. This brings the process to Fig. 6A, (61) shown in stack 100s which correlates to Fig. 6A designated (61) between the step (ii) and step (iii).

At page 63, line 18, change the paragraph to read as follows:

a22

The request for service via the UDP to the KDC in stack 200s is seen at location (xi-2) "P2" <u>Fig. 7A</u> which is further continued on Fig. 7B where the Kerberos server receives the request via the UDP port then processes the request and builds the response, then sends out the response via the UDP port to the NX server 13 which function is continued as "P1" which relates back to Fig. 7A at location (xi-1).

At page 63, line 28, change the paragraph to read as follows:

a23

Here, at Fig. 7A, step (xi-1)=P1, there is a KDC event, such that the message is processed to send back to the originator. Then proceeding upward in stack 200S, the Kerberos Support Library will build the message header with a Transaction_ID match after which there will be an insertion of the message in the queue for delivery to the requester at location (xi) "P". Now referring to Fig 7B, the NX_MCP environment shows the sequence at "P" (xi) which involves a queue insertion event, followed by an action to validate that the queue is active and to cause an "event" for monitoring the process.

At page 63, line 37, change the paragraph to read as follows:

Then, a call is made to (61) located on stack 100S (Fig. 7A) followed by the events shown on the upper part of stack 100s, where the COMS-event has "happened" and notes a message for MARC. Then, the message is passed to MARC with the appropriate header, after which MARC receives the unsolicited message. At this time, there is also a validation cycle to validate that the client is still valid. After this, MARC recalls the client dialogue number and if valid, forwards the dialogue number. If the dialogue number is invalid, then MARC will discard the message. This is followed by a call to position (62) of Fig. 7B whereby the client-server 13 is seen to transport the response to the client with a message as --- "your password has been successfully changed".

a24

At page 64, line 20, change the paragraph to read as follows:

a25

Thus, in summary, the asynchronous service request from MARC designated as "process C" is seen in Fig. 6A so that now referring to Fig. 4, the client service request is being routed to the MARC whereby MARC requests for Kerberos service and the Kerberos Support Library receives the request for service (d1, Fig. 4), will then select the "asynchronous" message choice <u>YES</u> at (e) which will then trigger the process "C", location (f2) which is then instituted at Fig. 6A, together with Fig. 6B.

At page 66, lines 29-32, change the paragraph to read as follows:

a26

The client server 13 (Fig. 16) operates within a Kerberos domain which is provided by the KDC Kerberos Server 20 having a key distribution center KDC 22, and a Kerberos administrator program 24 (K-ADMIN).

At page 66 line 37, through page 67 line 1, change the paragraph to read as follows:

As seen in Fig. 16, the ClearPath NX Server 13 has communication lines to the client 10 and also to [[a communications lines to]] the Kerberos Server 20. Within the ClearPath Server 13, there is indicated the COMS program 42 which works in communication with MARC 40 and also with the Kerberos Support Library 34. The MARC program 40 communicates with the Master Control Program (MCP) 60 which also has communication lines to the Encryption Library 32. The Kerberos Support Library 34 is connected with communication lines to the general security services application program interface 38.

a27

At page 74 line 19, change the paragraph to read as follows:

a28

At step 38i, there is a communication from the MCP 60 over to the COMS program 42 as was described in the co-pending U.S. patent application, Serial 09/026,746, now U.S. Patent 6,175,920 entitled "Expedited Message Control for Synchronous Response in a Kerberos Domain", which is incorporated herein by reference.

At page 74 line 32, change the paragraph to read as follows:

29

Then at step 40i, the communication management system COMS 42 recognizes this queued message as a message for MARC 40 if the message has a mutually agreed upon value in the first word of the Header. At step [[41]] 41i, COMS 42 then sends a message over to MARC 40 by calling a procedure and passing in a mutually known function code. Thus signals MARC 40 that this message is from Kerberos.